ISSN 0265-2919

R 80p 4

THE HOME COMPUTER ADVANCED COURSE

MAKING THE MOST OF YOUR MICRO

An ORBIS Publication

IR £1 Aus $1.95 NZ $2.25 SA R1.95 Sing $4.50 USA & Can $1.95

CONTENTS

Next Week

- **IBM is a household name. How much of an effect has this had on the marketability of the PC?**
- **We continue our in-depth reviews of disk drive systems, focusing this time on the BBC disk operating system.**
- **Repairing or upgrading your micro can be satisfying and save you money. In our Workshop series we look once more into the nuts and bolts of your micro.**

Your special offer binder order form will be with Issue 5.

Overseas readers: this special offer applies to readers in the U.K., Eire and Australia only.

COVER PHOTOGRAPHY BY PAUL CHAVE

Editor Jonathan Hilton; **Art Director** David Whelan; **Deputy Editor** Roger Ford; **Production Editor** Catherine Cardwell; **Staff Writer** Brian Morris; **Picture Editor** Claudia Zeff; **Designer** Hazel Bennington; **Sub Editors** Robert Pickering, Keith Parish; **Art Assistant** Liz Dixon; **Editorial Assistant** Stephen Malone; **Researcher** Helena Siedlecka; **Contributors** Lisa Kelly, Steven Colwill, Richard King, Martin Hayman, Henry Budgett, Steven Pizey; **Group Art Director** Perry Neville; **Managing Director** Stephen England; **Published by Orbis Publishing Ltd: Editorial Director** Brian Innes; **Project Development** Peter Brookesmith; **Executive Editor** Chris Cooper; **Production Co-ordinator** Ian Paton; **Circulation Director** David Breed; **Marketing Director** Michael Joyce; **Designed and produced by Bunch Partworks Ltd; Editorial Office** 85 Charlotte Street, London W1; **© APSIF Copenhagen 1984; © Orbis Publishing Ltd 1984: Typeset by Universe; Reproduction by Mullis Morgan Ltd; Printed in Great Britain by Artisan Press Ltd, Leicester**

HOME COMPUTER ADVANCED COURSE – Price UK 80p IR £1.00 AUS $1.95 NZ $2.25 SA R1.95 SINGAPORE $4.50 USA and CANADA $1.95

How to obtain your copies of HOME COMPUTER ADVANCED COURSE – Copies are obtainable by placing a regular order at your newsagent, or by taking out a subscription. Subscription rates: for six months (26 issues) £23.80; for one year (52 issues) £47.60. Send your order and remittance to Punch Subscription Services, Watling Street, Bletchley, Milton Keynes, Bucks MK2 2BW, being sure to state the number of the first issue required.

Back Numbers UK and Eire – Back numbers are obtainable from your newsagent or from HOME COMPUTER ADVANCED COURSE. Back numbers, Orbis Publishing Limited, 20/22 Bedfordbury, LONDON WC2N 4BT at cover price. AUSTRALIA: Back numbers are obtainable from HOME COMPUTER ADVANCED COURSE. Back numbers, Gordon & Gotch (Aus) Ltd, 114 William Street, PO Box 767G Melbourne, Vic 3001. SOUTH AFRICA, NEW ZEALAND, EUROPE & MALTA: Back numbers are available at cover price from your newsagent. In case of difficulty write to the address in your country given for binders. South African readers should add sales tax.

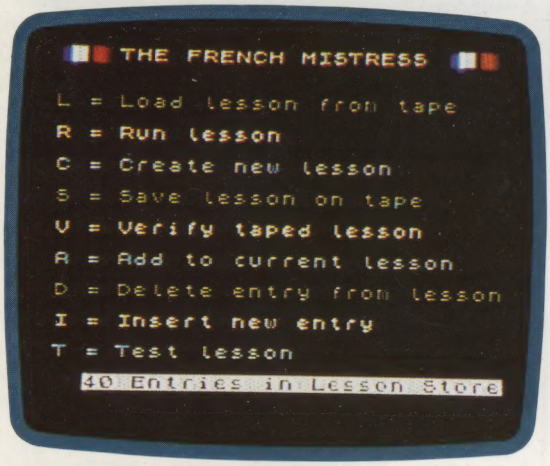
How to obtain binders for HOME COMPUTER ADVANCED COURSE – UK and Eire: Details of how to obtain your binders (and of our special offer) are in issue 5. EUROPE: Write with remittance of £5.00 per binder (incl. p&p) payable to Orbis Publishing Limited, 20/22 Bedfordbury, LONDON WC2N 4BT. MALTA: Binders are obtainable through your local newsagent price £3.95. In case of difficulty write to HOME COMPUTER ADVANCED COURSE BINDERS, Miller (Malta) Ltd, M.A. Vassalli Street, Valletta, Malta. AUSTRALIA: For details of how to obtain your binders see inserts in early issues or write to HOME COMPUTER ADVANCED COURSE BINDERS, First Post Pty Ltd, 23 Chandos Street, St. Leonards, NSW 2065. The binders supplied are those illustrated in the magazine. NEW ZEALAND: Binders are available through your local newsagent or from HOME COMPUTER ADVANCED COURSE BINDERS, Gordon & Gotch (NZ) Ltd, PO Box 1595, Wellington. SOUTH AFRICA: Binders are available through any branch of Central Newsagency. In case of difficulty write to HOME COMPUTER ADVANCED COURSE BINDERS, Intermag, PO Box 57394, Springfield 2137.

Note – Binders and back numbers are obtainable subject to availability of stocks. Whilst every attempt is made to keep the price of the issues and binders constant, the publishers reserve the right to increase the stated prices at any time when circumstances dictate. Binders depicted in this publication are those produced for the UK market only and may not necessarily be identical to binders produced for sale outside the UK. Binders and issues may be subject to import duty and/or local taxes, which are not included in the above prices unless stated.

SCHOOL DAZE

Educational computer software aimed at those sitting General Certificate of Education examinations is becoming more and more common. Almost invariably it is based on the question-and-answer method, and is constrained to the simplest kind of questions – those with absolutely definite unambiguous answers.

There are now numerous revision packages on the market and few make demands on the programmer. The task is a simple one of text presentation and monitoring a response from the user. Once this procedure is established the same structure can be used to accommodate a wide variety of subject texts and question-and-answer routines, with the object of supplementing or supplanting the text book. Though such programs are effective — and certainly cost-effective — it must be admitted that they can be rather dull. However, future generations of examination revision software will undoubtedly make use of the colour and graphics facilities now available on the majority of home computers to display visual material to illustrate the text.



Foreign Languages

The languages most commmonly taught to GCE standard are well supplied with revision software for a variety of machines, including those established in schools, such as the BBC Microcomputer and the ZX Spectrum, and those more commonly used as vehicles for games software, such as the Atari range.

One such group is available from Kosmos Software for both the Spectrum and BBC Model B on cassette. The French Mistress, the German Master and the Spanish Tutor are similar in their operating method, and all three come in two levels. Level A is composed of lists of words or short phrases (up to 59 characters long) in English and the subject language, in categories such as 'Family', 'Food', 'Living Creatures' and the like; while the Level B cassette progresses to adjectives and adverbs, and verb conjugations and tenses.

The user has the option of working from the foreign language into English, or vice versa, and also creating word lists of his own, which may be saved on cassette for later reference. The cassettes are split into 16 'lessons' each, the maximum length of any one of which is 250 entries. This limit also applies to user-created lists. The user can choose between the learning mode, in which a foreign word and then its English equivalent are displayed; a self-test, in which only one of the languages is displayed; and a timed accuracy test.

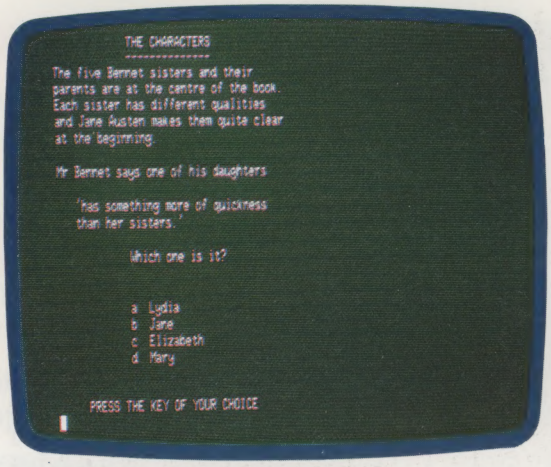
SOFTWARE COURTESY OF PILOT SOFTWARE

English Language

Teaching the native tongue is quite different from teaching a foreign language. The objective here is to teach one how to use the language to best effect. Here the computer can be useful insofar as rules of grammar can be applied. But the skills of writing précis, paraphrases, essays and compositions are matters of judgement rather than rule-based exercises and so do not readily lend themselves to computer-assisted revision.

Commodore's English Language is available on cassette and is based on material supplied by International Correspondence Schools. It forms part of a larger series for expanded Vic-20s, and concentrates on grammar and the use of words. The main menu offers a choice between Composition (which is in fact a mixture of exercises in word definitions and usage in the form of a multiple choice approach), spelling, grammar and comprehension.

The package is supported by a manual which explains the operating method with adequate examples but also provides text for the summarising and comprehension exercises.

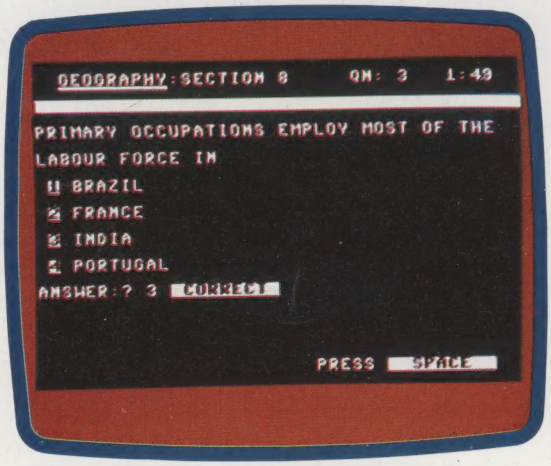


English Literature

Unfortunately for the software producers, English Literature syllabuses at both Ordinary and Advanced levels are based on particular works – and the list of their titles is a long one.

Sussex Software's Pride and Prejudice is an excellent example of the large amount of work that goes into the production of a package of this type. Written for Research Machines' 380Z (and hence, with some modification, potentially available for any other CP/M based computer) and the Commodore 3000 and 4000 series, the package is delivered on three disks, and concentrates on the characters, themes and story, as well as the social context.

This package, in common with other offerings in the fields of literature and history from this supplier, is claimed to extend to the CSE syllabus, but seems much more appropriate to GCE Ordinary and Advanced level students. While Sussex Software has been obliged to use multiple choice as its method of asking questions, each question is reinforced by a comprehensive explanation, and the reasons for arriving at the answer are given as fully as possible.



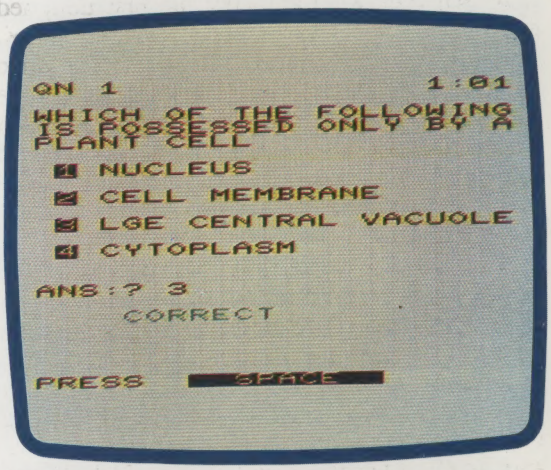
History And Geography

History, geography and economics demand interpretation of hard facts and are thus subjects that lend themselves well to computerised revision. Learning in these areas is largely a matter of rote, and it is in this field that computerised revision performs best.

Sussex Software has issued more than a dozen such packages; its main series is aimed at 380Z and Commodore 4000 users, while a parallel series is written for the Sinclair Spectrum, marketed under the Akadimias imprint. These latter programs should also become available for the BBC Microcomputer during 1984.

Commodore also has a history package, The History of the 20th Century, for the Commodore 64. Published jointly with Ivan Berg Software (which also co-publishes the titles available for the Vic-20), with editorial material supplied by Hodder and Stoughton, this package is a comprehensive revision guide to the political, social and economic history of the period, laid out in eight segments.

The package is available on cassette from Commodore dealers and independent software retailers and costs around £10, as do most of the other packages described here.



LIZ HEANEY

Mathematics And Sciences

At GCE level there is no latitude for personal interpretation in mathematics and science. These subjects are perhaps the easiest to present within the constraints of a computerised revision program. There are a vast number of products available in this section of the market, covering both general and specific aspects of the subjects (geometry or trigonometry, for instance, within the mathematics curriculum).

Commodore's mathematics packages, which are in two parts, are good examples of a general mathematics revision course. Similar software is available for most of the microcomputers normally found in the home. It is also worth mentioning Rose Software's series of titles for the Spectrum, and SciCAL's for the BBC Microcomputer. Because much mathematical and scientific material is easily illustrated by simple diagrams, it is in this area of educational programs that computer graphics made their first appearance.

Revision material is available that covers the common scientific subjects (chemistry, physics and biology) 'across the board' or in individual aspects of a particular syllabus – heat and light, for example, or physiology. Packages of this nature are the mainstay of educational software and are available from a wide variety of sources.

GOLDEN OLDIES

The Atari 810 disk drive has been on the market for some time now and many of its features are outdated. The storage capacity is limited and the speed of operation is also comparatively slow. However, it does have a comprehensive range of disk commands. Explanations of the most frequently used commands and routines are documented here.

When the Atari 400 and 800 computers were launched they were arguably the first designed specifically for home use. Atari almost immediately produced the 810 single disk drive system to augment the capabilities of the computers, but because of the high cost of each drive (now £299), the system has not proved popular. However, with the advent of the compatible Atari 600, it is worth looking afresh at the 810 disk drive.

The 810 uses single-sided, single density $5\frac{1}{4}$in floppy disks. It is connected to the computer via the special parallel I/O port. Up to four 810s can be 'daisychained' one to another, numbered 1 to 4 according to the settings of a drive code switch on the rear of each drive. Although the 810 contains its own microprocessor it cannot be considered a true intelligent drive, as part of Atari DOS II, the Disk Operating System currently supplied, must be loaded into RAM before the drive can be accessed. DOS takes up approximately five Kbytes of user RAM, leaving barely eight Kbytes of RAM free on 16-Kbyte machines such as the 600. However, in the near future, 48-Kbyte RAM boards will be available for the 600 to increase the RAM to 64 Kbytes.

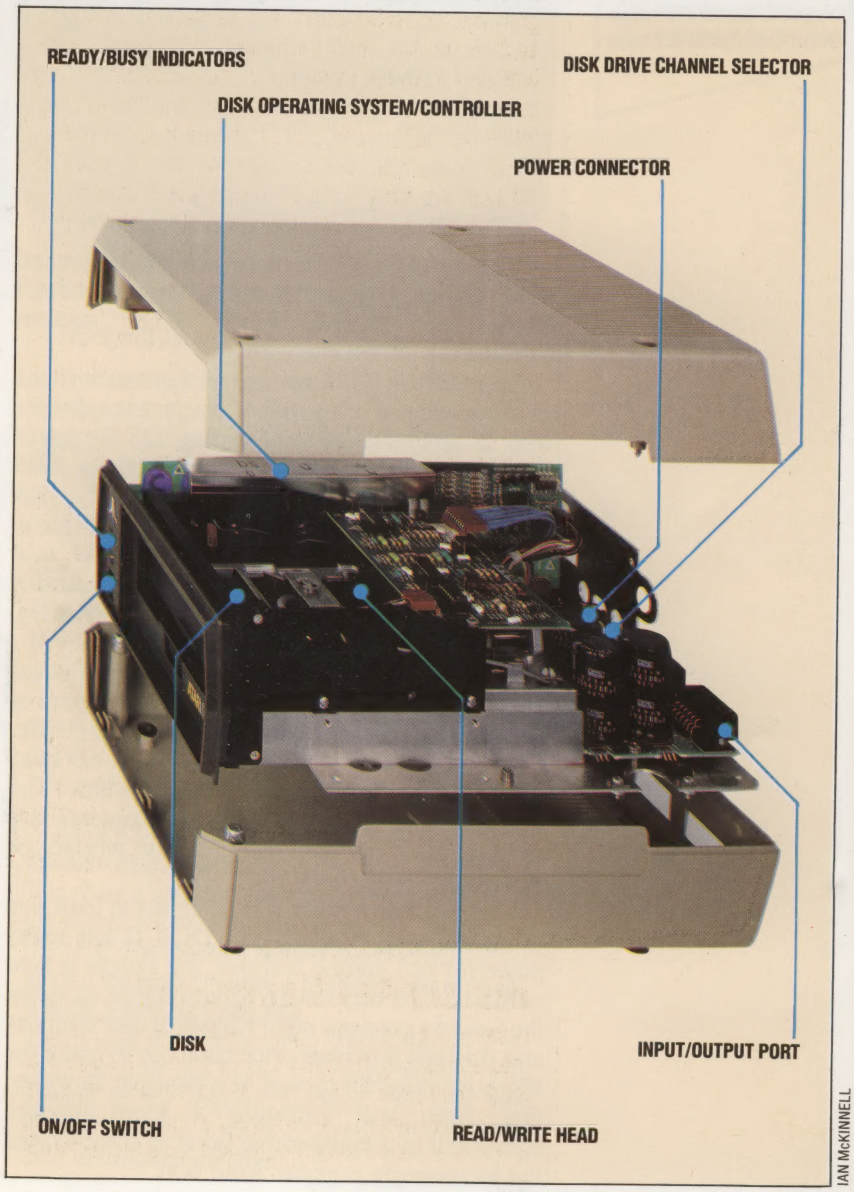
Atari DOS is supplied with the disk drive on a master disk and consists of three separate but associated files. These are: DOS.SYS, which contains the File Management System (FMS) and the command instructions resident in RAM while in use; DUP.SYS, a disk utility file that holds the DOS menu and some DOS command instructions; and AUTORUN.SYS, which contains a file that on command is automatically loaded into RAM and executed to call the DOS menu and RAM-resident portions of the DOS.

To access DOS with the BASIC cartridge inserted, the drive must be switched on before the computer is powered up and the master disk is inserted. Switching the computer on then 'boots'—or more specifically loads—part of DOS.SYS into RAM. To call the 15-option DOS menu you have to type DOS (RETURN). If no cartridge is inserted, the DOS menu is called automatically from boot.

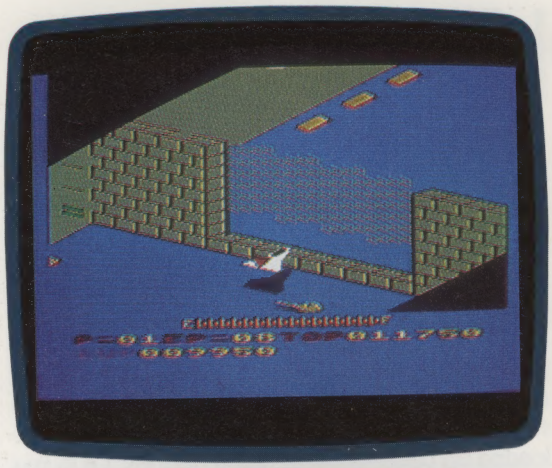
The DOS menu provides 15 disk management command options selected with the letters A to O. The menu system offers a simple method of disk management but it has a major drawback in that it overwrites a portion of user memory, destroying any program or data currently stored. This means that a program being edited or written must be saved to disk before calling the menu and reloaded on completion of menu options. DOS has a facility by which this can be carried out automatically if required, but this still seems a clumsy arrangement.

DOS includes an automatic verification of data stored with every write command, which limits the data transfer rate to 2.4 Kbytes per second. If greater speed is required, the automatic

The Atari 810
The greatest drawbacks of the Atari 810 are its serial interface, which makes the drive rather slow in operation, and its limited storage capacity of 88 Kbytes per disk. A rather sophisticated disk operating system, however, does help to counter these shortcomings



IAN McKINNELL



In The Cockpit
Zaxxon is just one of the very successful range of games that Atari has produced for its home computers. The player is the pilot of a jet fighter, in combat with other aircraft and pressing home a ground attack. The screen display becomes the aircraft's windscreen

Blue Max
A second dogfight game from Atari takes the player back to a rather different era. Blue Max, available on disk, cassette or cartridge, places you at the controls of a World War One biplane fighter. While it doesn't pretend to be a flight simulator, the game's main strength is in the quality of its graphics

COURTESY OF SOFT

verification can be disabled with the command POKE 1913,80, giving a data transfer rate of 4.8 Kbytes per second. POKE 1913,87 restores verification.

The FORMAT DISK command formats a blank disk in a selected drive with 40 tracks, each divided into 18 sectors capable of storing 128 Kbytes of data apiece, with three bytes reserved for FMS use. DOS allocates eight sectors for the disk directory and one sector for the Volume Table of Contents (BAM equivalent) and reserves four sectors for DOS use, giving a total of 707 sectors × 125 bytes, or 88,375 bytes (about 86 Kbytes) available for storage.

The standard LOAD, SAVE and allied BASIC commands can be used to store data as program or data files by specifying disk as the storage device. Files can also be stored that can be read sequentially or randomly, one byte at a time. The directory and Volume Table of Contents are automatically updated as files are written or changed.

The 810 disk system was designed at least five years ago and though Atari DOS II has many powerful and unique features, the system is now outdated. The very low 86 Kbytes of disk capacity, the loss of a precious eight Kbytes of user memory and the high purchase price combine to make the 810 poor value for money. It is possible, however, that Atari will introduce a more modern equivalent at a competitive price to complement the increasingly popular Atari 600.

Atari Disk Commands

When manipulating files a standard file specification is generally used. This is:

COMMAND "DN:FILENAME.EXT"

where COMMAND is the DOS command; N is the drive number (1-4 optional on single drive); the FILENAME contains up to eight characters to identify the file (first must be alphabetic); and .EXT is the optional file extender (this can be used to indicate the type of data stored).

Where this file specification is required it is referred to as FSP. It is also possible to use 'wildcards' with some commands where '?' can be used to substitute for a single character and '*' can substitute for any number of valid characters after the '*'.

To select an option on the DOS menu, type the appropriate letter and press (RETURN). In all cases FNM means D1:FILENAME.EXT.

A. DISK DIRECTORY
(RETURN) displays a list of all file names contained on the disk in drive 1, the extender and number of sectors for each file.

B. RUN CARTRIDGE
This option returns control of the computer to the cartridge inserted, usually BASIC.

C. COPY FILE
This copies a file from one disk to another or to the same disk under a different name. For example:

D1:FILENAME.EXT, D2:FILENAME, EXT

copies FILENAME.EXT from drive 1 to drive 2. Similarly:

D1:FILENAME.EXT,D1:FILENAME.BAK

makes a back-up copy of a file on the same disk. The file name must be different in some manner and in this case the extender is changed.

D. DELETE FILE(S)
FNM (RETURN) deletes a specified file or files. If all files are to be deleted use '*' instead of file name and extender.

E. RENAME FILE
This renames a specified file and updates the directory. For example:

D1:OLDNAME, NEWNAME

Wildcards can be used to change the extenders of a group of files.

F. LOCK FILE
FNM (RETURN) software-protects a file from being amended or deleted unless the disk is re-formatted. Files thus protected are preceded by '*' on a displayed directory.

G. UNLOCK FILE
FNM (RETURN) unlocks the specified file or all relevant file names if used with wildcards.

H. WRITE DOS FILES
Follow the instructions displayed to save DOS to a formatted disk.

I. FORMAT DISK
Follow instructions to format a disk.

J. DUPLICATE DISK
Follow the instructions to copy an entire disk from drive to drive or, by copying the contents of the disk into memory, to the same drive.

K. BINARY SAVE
FNM,SSSS,EEEE (RETURN) saves the contents of a specified area of memory, usually a machine code program. SSSS is the start address and EEEE the end address in four-digit hexadecimal.

L. BINARY LOAD
FNM (RETURN) loads a file saved with BINARY SAVE back into the locations from which it was stored.

M. RUN AT ADDRESS
On display prompt enter address at which to execute a BINARY LOADed file in four-digit hexadecimal. (RETURN) executes the program.

N. CREATE MEM. SAV
Follow the instructions to create a file called MEM.SAV. When the DOS menu is called, DOS automatically saves the contents of memory that would be overwritten by the menu and reloads it when RUN CARTRIDGE is selected.

O. DUPLICATE FILE
FNM (RETURN) then follow the instructions to copy files from one disk to another on a single drive. Wildcards are permissible.

The other commands controlling program files and data files are:
SAVE LOAD LIST ENTER RUN OPEN# CLOSE# PRINT#INPUT# NOTE# POINT# PUT# GET# STATUS# X10

Program Files:

SAVE FSP
This writes the specified program to disk in a 'tokenised' form.

LOAD FSP
This reads the specified tokenised program into user memory from the bottom of memory upwards.

LIST FSP,LN1,LN2
This stores a BASIC program IN ATASCII form (Atari's version of standard ASCII). Specifying FSP alone stores the whole program. LN1 and LN2 represent line numbers and can be used to specify the start and end line numbers of a portion of a program to be stored. Used in conjunction with ENTER to merge programs.

ENTER FSP
This reads the specified file, previously stored using LIST, into user memory and merges it with a program already held in memory. If there are duplicate line numbers, the lines contained in the newly read file take the place of those in memory.

RUN FSP
This reads the specified tokenised program into memory and automatically runs it.

Data Files:

OPEN#
This command controls access to special communication channels called I/O Control Blocks (IOCB) and links them to an appropriate device, in this case a disk drive and FSP, under specified conditions, as follows:

OPEN#IOCB,AC1,AC2,FSP

where IOCB is the I/O channel (1-5); AC1 is the auxilliary code 1 (specifies type of I/O operation according to a table in DOS manual); and AC2 is always 0 for disk.

The following commands relate to IOCBs OPENed in the manner previously explained.

CLOSE# IOCB
This releases the specified IOCB from the I/O conditions set above. CLOSEd IOCBs cannot be accessed.

PRINT#
This writes numeric (X,Y) or string (A$) data to the specified IOCB, for example:

PRINT#,X,Y or PRINT#, IOCB,A$

INPUT#
This reads numeric or string data from the specified IOCB, for example:

INPUT#IOCB,X,Y or INPUT#IOCB,A$

NOTE#
This is set before storing data with PRINT#. It provides a record of the sector number and byte number where the next byte is to be stored on disk. The resulting list can be stored as a table in another file to enable data to be accessed randomly, a byte at a time if required, by POINT#. For example:

NOTE#IOCB,A,B

where S is the sector number (1-719) and B is the byte number (0-124).

POINT#
This reads the specified byte of data, previously stored using NOTE#, into user memory thus:

POINT#IOCB,A,B.

PUT#
This writes a single byte to the specified IOCB. For example:

PUT#IOCB,N

where N = 1 to 255.

GET#
This reads a single byte stored by PUT# with:

GET#IOCB,N

STATUS#IOCB, ERROR
This gives a specified variable, in this case ERROR, the current error number for the last I/O operation on the IOCB. The resulting number can then be checked against the table of errors in the Atari DOS manual.

X10 CN,#IOCB,AC1,AC2,FSB
This is used to duplicate some of the functions of the DOS menu by use of the Command Number (CN) relating to the required function. The Atari DOS manual contains a list of Command Numbers and their associated functions.

PRICE DIFFERENTIAL

Since 1973, the cost of a computer has dropped 500 fold per bit of memory. If the Rolls Royce car had been developed at the same rate as the computer — while achieving a similar level of price reduction — today's vehicle would cost in the region of £1.50 and average about one million miles to the gallon

PROGRAMMER'S LOGIC

So far in this series on logic we have looked at the design of *hardware* using logic gates, but the logical operators AND and OR are also useful *software* tools within programs. Most BASIC dialects and machine code instruction sets include AND and OR in their commands.

These two logical operators have several uses in both machine code and BASIC. The most familiar use of AND and OR is to relate two or more statements within a conditional statement. For example, try predicting the outcome of this BASIC program:

```
10 FOR I=1 TO 5
20 FOR J=1 TO 5
30 IF I=3 AND J=2 THEN PRINT I,J
40 NEXT J
50 NEXT I
60 END
```

The program will run through the pair of nested loops but will print out the values of I and J only if I=3 and J=2. This program will therefore print on the screen the following result:

```
3 2
```

OR can be used in much the same way. If we amend line 30 to read:

```
30 IF I=3 AND J=2 OR J=4 THEN PRINT I,J
```

The following output is produced:

```
1 4
2 4
3 2
3 4
4 4
5 4
```

The computer carries out the AND operation in priority to the OR operation. I and J will be printed out if either I=3 and J=2, or if J=4. The order of priority can be changed by the use of brackets. What will be the output from the program if line 30 is again amended to:

```
30 IF I=3 AND (J=2 OR J=4) THEN PRINT I,J
```

ISOLATING BITS IN A REGISTER

Many home computers use special registers to control various machine functions. Each bit within such a register may control a different aspect of that operation. For example, on the Commodore 64 there is an eight-bit register that controls the switching on and off of sprites. Each bit in the register relates to one of the eight sprites available. If any bit in the register is set to a one then the sprite that it controls is visible on the screen. If the bit is set to a zero then the sprite is switched off and cannot be seen. Using BASIC it is a simple matter to switch on any combination of sprites by working out the required eight-bit binary number and POKEing its decimal equivalent into the register. This method, however, doesn't take account of the state of the register prior to the POKE command and may result in switching off sprites that were previously on. The solution to this problem is to develop a technique that will allow the programmer to isolate the bits that are needed to be changed without altering any of the others.

In order to demonstrate this technique let us assume that originally sprites 0, 1, 5 and 6 are turned on. The register controlling the switching will look like this:

Sprite Number	7	6	5	4	3	2	1	0
Corresponding Bit	0	1	1	0	0	0	1	1

Let us now switch on sprite 4, by PEEKing the register, ORing the contents with 16 (00010000 in binary) and POKEing the result back.

Original Byte	0	1	1	0	0	0	1	1
ORed With	0	0	0	1	0	0	0	0
Gives New Byte	0	1	1	1	0	0	1	1

Using the BASIC command POKE reg,PEEK(reg) OR16 we can now turn on bit 4 in the register. To turn bit 4 off again we must PEEK the register and AND its contents with 239.

New Byte	0	1	1	1	0	0	1	1
AND	1	1	1	0	1	1	1	1
Original Byte	0	1	1	0	0	0	1	1

Notice that the number 239 can be quickly calculated by subtracting 16 from 255. Using the BASIC command POKE reg,PEEK(reg)AND239 we have restored the register to its original state.

These techniques are more widely applied in machine code programming where altering the state of control registers may form an important part of the program.

NOSE-OPERATED KEYBOARD

The MATE (Memory Assisted Terminal Equipment) system was developed at Essex University, England, in 1978 on a Vector 111 computer. This system was developed by a severely spastic programmer, Geoff Busby, who could manipulate the keyboard only with his nose.

MATE has a special keyboard that does not require a shift key to be held down at the same time as another key for some characters, and stores a database of words that can be input by a single keystroke without typing all characters

Answers to Exercise 3

1a) $A.(\overline{A}+\overline{B})$

$= A.\overline{A} + A.\overline{B}$ (distributive law)

$= A.\overline{B}$ ($A.\overline{A}=0$)

b) $X+Y.(X+Y)+X.(\overline{X}+Y)$

$=X+Y+X.(\overline{X}+Y)$ (relation 5)

$=X+Y+X.Y$ (relation 6)

$=X+Y$ (absorption)

c) $P.Q+\overline{P}.Q+P.\overline{Q}$

$=P.Q+\overline{P}.(Q+\overline{Q})$ (distributive law)

$=P.Q+P$ ($Q+\overline{Q}=1$)

$=\overline{P}+Q$ (dual of relation 6)

d) $\overline{X+Y.\overline{Z}+\overline{Z}.Y}$

$=\overline{X}.\overline{Y.\overline{Z}}.\overline{\overline{Z}.Y}$ (de Morgan)

$=X.Y.Z.(Z+\overline{Y})$ ($\overline{X}=X$, de Morgan)

$=X.Y.Z.Z+X.Y.Z.\overline{Y}$ (distributive law)

$=X.Y.Z+0$ ($Z.Z=Z$, $Y.\overline{Y}=0$)

$=X.Y.Z$

2) The truth table for the alarm system is:

INPUTS			OUTPUTS
A	B	S	Alarm
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

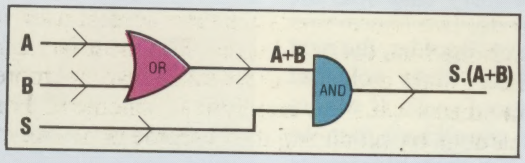
Alarm $= \overline{A}.B.S+A.\overline{B}.S+A.B.S$

$=\overline{A}.B.S+A.S.(\overline{B}+B)$ (distributive law)

$=\overline{A}.B.S+A.S$ ($\overline{B}+B=1$)

$=S.(A+\overline{A}.B)$ (distributive law)

$=S.(A+B)$ (dual of relation 6)



3) If the three switches are X,Y and Z and the hall light is P then the truth table is:

INPUTS			OUTPUTS
X	Y	Z	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$P=\overline{X}.\overline{Y}.Z+\overline{X}.Y.\overline{Z}+X.\overline{Y}.\overline{Z}+X.Y.Z$

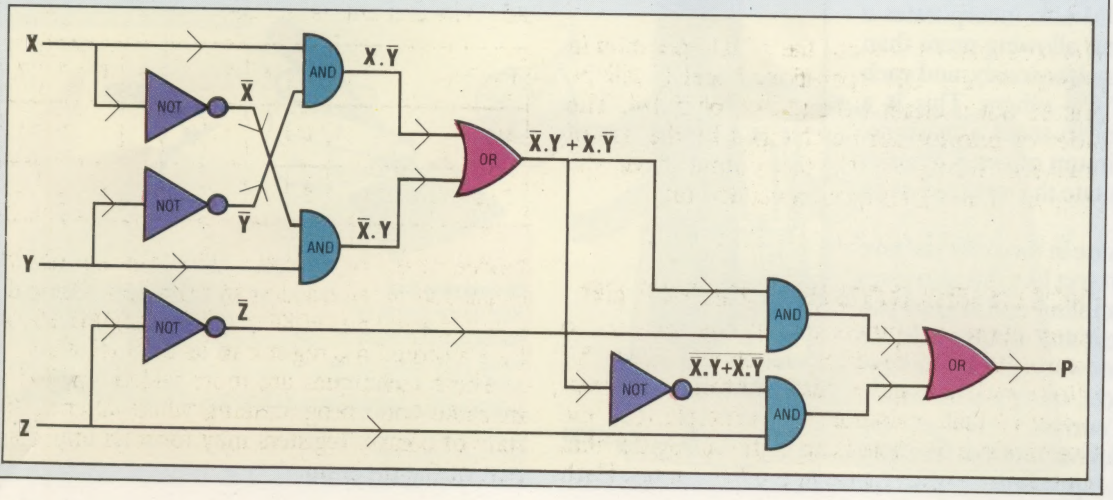
$=Z.(\overline{X}.\overline{Y}+X.Y)+\overline{Z}.(\overline{X}.Y+X.\overline{Y})$ (distributive law)

$=Z.\overline{(\overline{X}.Y+X.\overline{Y})}+\overline{Z}.(\overline{X}.Y+X.\overline{Y})$ (de Morgan)

4) The given truth table shows that the question "Do you tell the truth?" is of little use to us because both a liar and a truth-teller will give the same reply. The truth table has the same form as the function $X.Y+\overline{X}.\overline{Y}$, which simplifies to Y. That is, the answer is dependent on only one variable, not two, so the question does not differentiate between liars and truth-tellers. However, if we ask the question, "Do pigs have wings?" then the table is:

		POSSIBLE ANSWER	
		YES	NO
POSSIBLE IDENTITY OF RESPONDENT	LIAR	1	0
	TRUTH TELLER	0	1

and this is the truth table for the function $X.\overline{Y}+\overline{X}.Y$, which is also an Exclusive-OR table. This question enables us to identify the respondent.



SOCKET TO ME

Making up leads and connectors is just the first step in basic electronics assembly work. The next stage is de-soldering and component replacement. Our exercise this time involves removing a ROM chip and replacing it with a Zero Insertion Force socket, into which the memory chip will fit.

Now that we have learned how to solder, the next step is to find out how to undo soldered connections, neatly and without mess. If we are dealing only with cables and connectors, then we need not worry too much — it is probably simpler to cut back the cable, throw away the plug and start afresh with new components. But what happens when we come to replace or re-locate a chip? Even simple transistors have three pins or terminals. In order to free a transistor from its printed circuit board, we need to heat all three simultaneously to the melting point of solder, so that the transistor can be pulled clear; or we must clear each pin of solder in turn. And if the notion of releasing three pins at once is daunting, how about the 40 pins of the average eight-bit microprocessor?

REPLACING THE CHIP

Rather than simply choosing a chip at random to be replaced, or just adding RAM chips to expand a machine's capacity, we're going to take on a slightly more ambitious project: replacing a ZX81's BASIC ROM with a socket, leading a ribbon cable outside the case to a piece of Veroboard or a Veroblock, and installing a Zero Insertion Force socket to take either the ROM we have extracted, or any other. The reason for choosing the ZX81 as the subject for this particular exercise is that FORTH is available as a replacement resident language. In addition to the new language, the ROM also incorporates a multi-task operating system, allowing more than one program to operate simultaneously, and each entirely independently of the others. This is a remarkable achievement in so small a machine, though it does need a minimum of two Kbytes of RAM, which might necessitate the addition of an extra memory module.

As well as being an exercise in the skills we have so far examined, a small project like this also gives practice in component handling and shows you how necessary it is to be neat and accurate . In the next instalment of the series we shall look at ways of testing the finished article, using a multimeter. This device measures current, voltage and resistance, and is an invaluable tool for checking and testing circuits and components.

Recipe

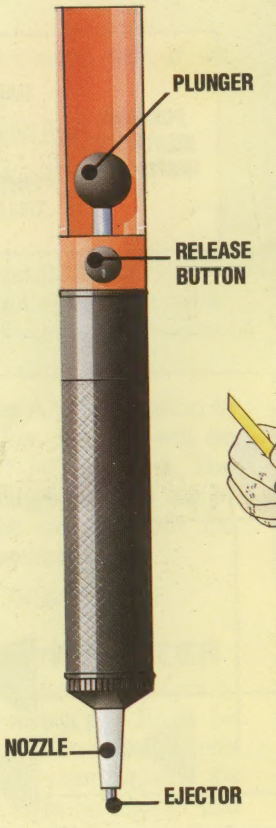
If ever a chip in your home computer is to be replaced, or, as in our example, simply relocated, or if a new one is to be added, it is an excellent practice to install a chip socket to carry it. Socketed chips can be replaced in a few moments; and if replacements are going to be fairly frequent, there are sophisticated chip holders available, called ZIF (Zero Insertion Force) sockets, which minimise the risk of bending a pin. In addition to the tools we described earlier (see page 44), you will need either a reel of de-soldering braid or a de-soldering tool. The exercise we have chosen calls for the removal of the BASIC ROM from a ZX81, and replacing it with a standard socket from which a ribbon cable will be led outside the machine to a ZIF socket. The objective is to replace ZX81's BASIC with David Husband's multi-tasking FORTH-in-ROM — while giving the user the option of returning to BASIC at a future date. If you wish to work through this example you will also need a piece of Veroboard and a length of 28-way ribbon cable, both of which are available from any electronics component supplier, and a piece of polystyrene

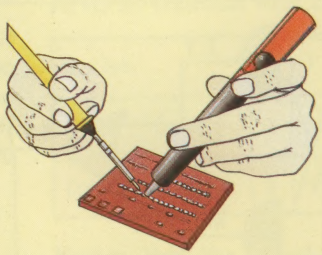
Locating The ROM

The first step is to open the ZX81's case and locate the BASIC ROM. The case is held together by five self-tapping screws, three of which are located under the stick-on rubber 'feet' on the underside. The pad that does not conceal a screw is the one closest to the EAR and MIC sockets. Peel the other three pads off carefully and remove the three cross-head screws underneath, and then the other two similar screws in plain sight. Lift off the underside of the case to reveal the bottom of the printed circuit board (PCB). This can then be released from the case by means of the three visible cross-head screws, two adjacent to the edge connector, the other near the heat sink (the aluminium plate beneath the keyboard). Turn the PCB over. The ROM chip we're looking for is located above and slightly to the left of the keyboard connectors

De-Soldering Tools

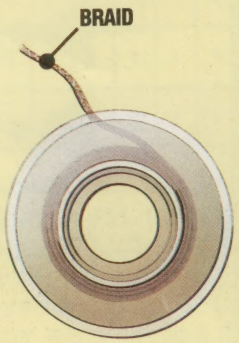
There are two proprietary aids to de-soldering. Least expensive in the short term is 'solder wick' — very fine copper wire braided into a tape and impregnated with flux, which relies on the phenomenon of capillary action (a function of surface tension which causes liquids to climb up narrow tubes) to draw the melted solder up into itself, just as a fabric wick will deliver fuel to the burner of a lamp or heater. Braided solder wick comes in a variety of widths, and the size relates very directly to the amount of solder to be removed. It is disposable, and cannot be reused. It costs around £1 for a five foot (1.5m) length. The second method is much more satisfactory, and consists of a device similar to a tiny spring-loaded bicycle pump, but working in reverse — sucking instead of blowing at the tip. They cost around £7 each, but of course have a very long life, and are much quicker in operation than braid. As you will see, one of these two methods is essential to the successful removal of components.

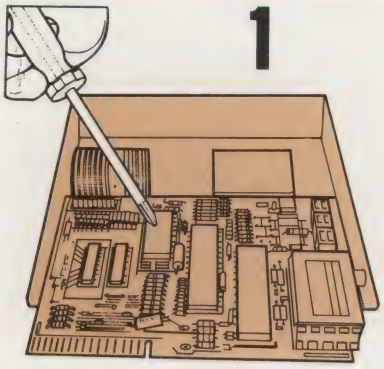


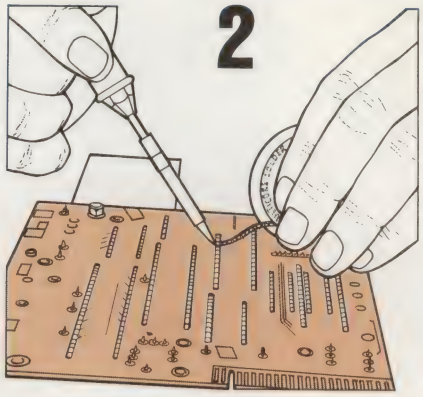


De-Soldering Tool

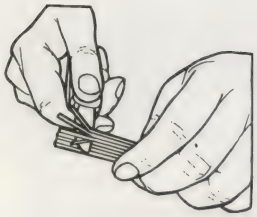
Heat up the joint to be released with the iron until the solder runs, apply the de-soldering tool, press the release button, and the liquid solder will be sucked up into the body of the tool



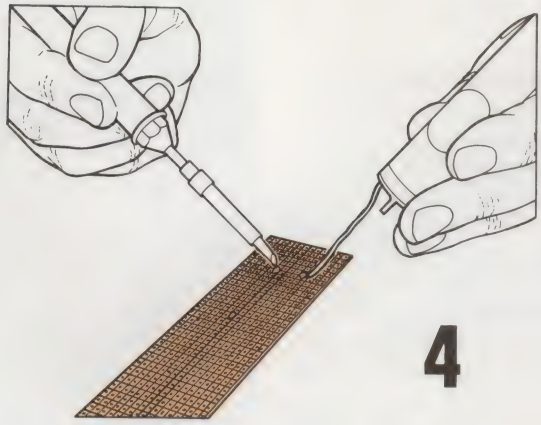




Detaching The ROM
When de-soldering, as with making joints, the secret is to apply sufficient heat. Press the solder braid onto the joint with the tip of the soldering iron until the flux in the braid runs — you will probably see a little blue smoke. You will see the solder braid sucking up the solder. Cut off the end of the braid regularly — each small section soon becomes saturated

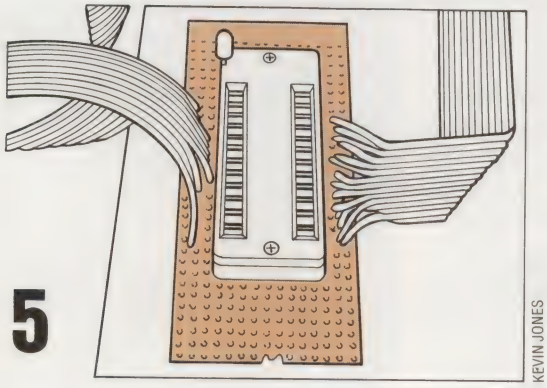


Ribbon Cable
Ribbon cable is available in a variety of widths: we need 28-way (i.e. 28 separate cables) or two strips of 14 way. Strip one end of each of the 28 cores for one centimetre ($\frac{1}{2}$ in.) and tin each core. Working from the end marked with a small semi-circular cut-out, solder the ribbon cable in place down both sides. All chips have these cut-outs, or sometimes dots, to show which way they should be aligned



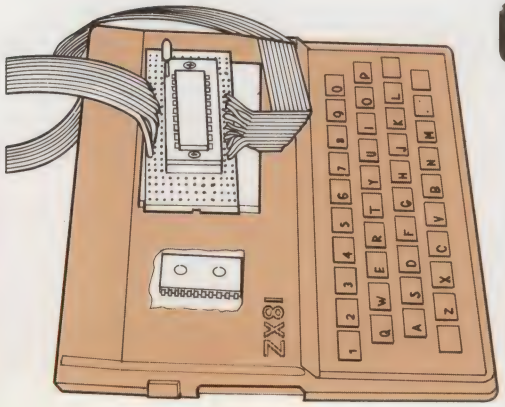
Connecting Up
Cut the two pieces of ribbon cable to length, and bare and tin the 28 cores. Then, making quite sure that the cores from the marked end of the area that held the chip go to the same end of the socket, solder them into the Veroboard. Pay particular attention to ensure that none of the cores becomes crossed

Veroboard And Veroblock
These two proprietary products are designed to stand in for printed circuit boards. Veroboard is a rigid plastic sheet, drilled in a standard matrix pattern, with copper strips forming each row of the matrix. The contact between each connection point can be broken by cutting through the copper strip, and connections can be made between the rows. Veroblock is a more sophisticated version of the same thing, but encapsulated in a plastic block with solderless connections



Mounting The Socket
For this purpose, Veroboard is the most appropriate medium. Fifteen contact points wide by around 25 cms (10ins.) long, it can be broken to an appropriate length between the fingers. Place the chip socket so that it straddles the central insulation, and solder one corner pin to hold it securely, then work your way down each side in turn, making sure that each joint flows cleanly

WARNING
Your home computer's guarantee — if still in force — may be rendered null and void if anyone other than the manufacturer or his appointed agent opens the case



Finishing Off
When all the connections are made, check each one, both on the computer's own PCB, and on the newly created daughter board, to see that each is separate from all the others. When working in miniature like this, with many connections close together, connections may become joined or 'tracked', which will lead to short circuiting. Check the connections with a magnifying glass, and if in doubt, score through the gap with a scribe or other pointed metal instrument

HARD COPY

Nearly all computer owners need to produce a hard copy print-out at one time or another. For business purposes an expensive letter-quality printer such as a daisy wheel is almost essential, but for the average home computer owner, who usually wants nothing more than a program listing, there are cheaper alternatives.

The means to produce a printed copy of a program listing or piece of text is high among any computer owner's priorities. Unfortunately, the cost of a good dot matrix printer may be two or three times the original price of the computer, while a daisy wheel device is certainly out of reach. Low-cost alternatives do exist, however, in the form of non-impact printers.

These printers take their generic name from the fact that they don't hammer needles or shaped pieces of metal through a ribbon to leave their mark on a piece of paper. The original non-impact printers were developed for cash registers and portable terminals such as the famous Texas Silent 700 series and used heat to develop an image on special, treated paper.

Just as an impact matrix printer has a vertical column of needles in a head that moves horizontally across the page, the thermal printer has a column of heating elements. As each dot is required, the corresponding element is rapidly heated and the minute area of paper under the element changes colour. The contrast between the dotted image thus produced and the unmarked background is good enough for most amateur uses but the letters hardly rate as high-quality. Indeed, the most endearing feature about these devices is their almost total silence of operation.

Apart from the Texas Instruments terminals, thermal printers were also adopted by Apple, which produced a small printer for its computer called the Silentype, and Mattel, which offers a similar unit for use with its Aquarius home computer.

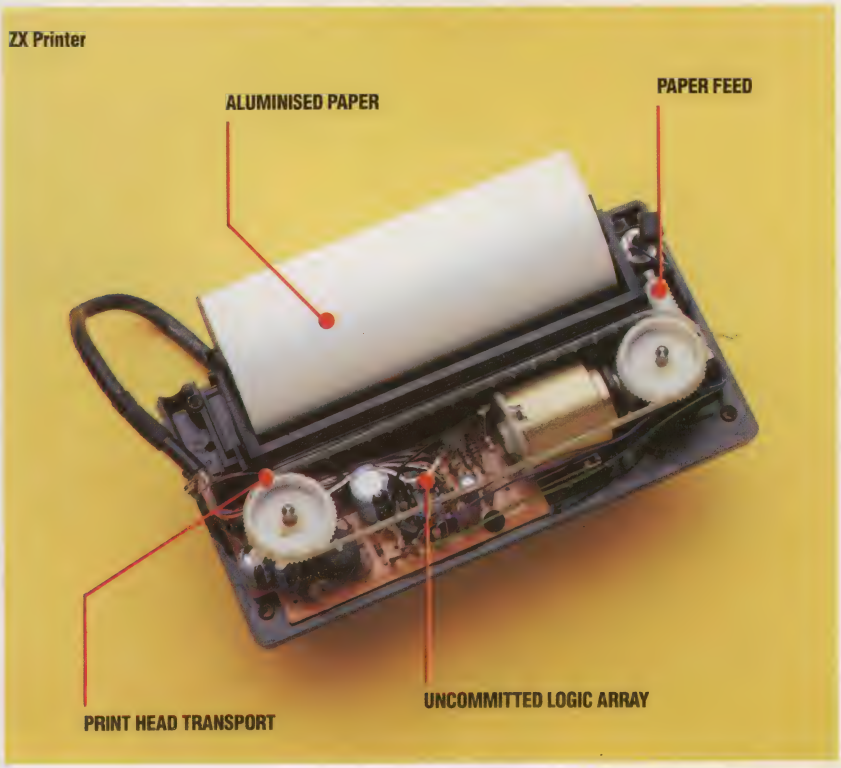
In contrast, the mechanism used in the electrostatic printer is moderately noisy and, apart from an early Centronics example called the Microprinter P1, it failed to gain general acceptance — until, that is, Sir Clive Sinclair adopted the system for the ZX Printer, which has sold in huge numbers as a dedicated peripheral for the ZX81 and ZX Spectrum.

The principle of electrostatic printers is that a single-wire printing head is dragged across specially coated paper. For every dot needed to build up a character, a spark is generated by the

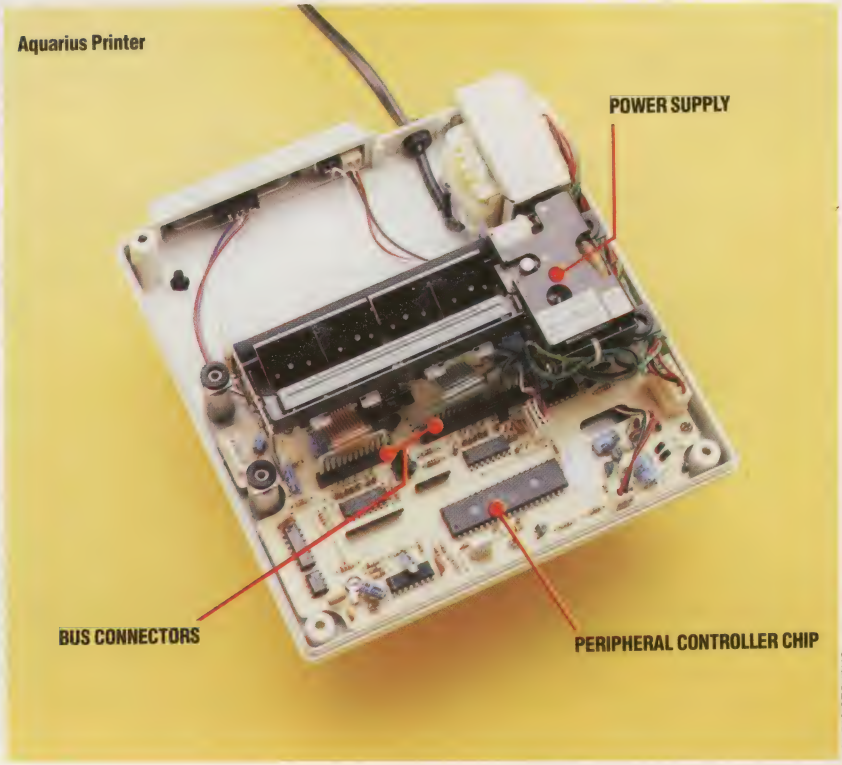
ZX Printer
Introduced as an output device for the ZX81, the ZX Printer is also directly compatible with the Spectrum. It offers an inexpensive method by which home computer users can obtain printed results and lists of programs. Though its low initial cost is attractive, its reliance on electrostatic paper leaves it at a distinct disadvantage

Aquarius Printer
Aquarius chose the other low cost operating method for its home microcomputer — thermal printing, though an inexpensive four-colour printer, which uses ball-point pens, is also available. Like the ZX Printer, it suffers through having to rely on special paper

ZX Printer



Aquarius Printer



CHRIS STEVENS

printer, which burns away the thin metallic coating to reveal the black backing paper. Sinclair improved the system by using two heads on a continuous belt, but it still takes eight passes of a head to create each row of characters. Fortunately, the ZX Printer has to print only 32 characters on each row so the speed is acceptable.

The main disadvantage of both types of printer is that they use special paper. Such materials tend to be expensive and are available only in rolls, which makes storage difficult. With the thermal printer it is essential to buy the correct grade of paper; otherwise the image will not develop properly. It also fades with time or exposure to heat. Electrostatic paper is even more delicate and if handled with damp or sweaty hands the image will blur and fade away as the coating dissolves. In both cases the best way to ensure a good, long-lasting image is to take a photocopy. Surprisingly, the silver-coloured paper photocopies very well.

These drawbacks notwithstanding, both types of printer offer the makers of the smaller home computer a way to provide some form of printing system for their machines that they would otherwise have to do without. The dox matrix method of printing allows direct copies to be made of the screen display, so both text and graphics can be produced at no extra cost, and the relatively low quality of the final image doesn't really matter to the user.

Recently, however, the dominance of these printing systems has been challenged by the four-colour pen plotters supplied by Tandy and Sharp. These can produce excellent, fully formed character sets and superb line graphics on ordinary roll paper for roughly the same cost as the thermal or electrostatic devices. As printing technology develops, it is still the non-impact device with its simple mechanism and limited facilities that fills the printing requirements of the low-cost home computer market.

Big Brother
Of all the implementations of thermal printer technology, few can be quite so elegant as the Brother EP-22. For around £170, less than many of the cheapest printers of any type, you not only get a 75-column thermal printer but a portable typewriter into the bargain. Designed to operate with thermal paper and as a normal impact printer with a ribbon fitted, the unit can store up to 2,000 characters in its battery-powered memory, allowing letters or memos to be created on the move. The electronic memory doesn't really support word processing as such but you can correct any character in the last 16 entered, as the output goes to a built-in LCD display before being typed onto the paper. The quality of the output isn't up to that of a real 'printer'— lower case letters lack true descenders for example—but as a portable printing companion to computers such as the Epson HX-20 and the Tandy Model 100 it performs a function that would be difficult to achieve with any form of impact printer

IAN McKINNELL

Tandy Thermal Printer
More adaptable than the other two low-cost printers we have described, the Tandy TP-10 Thermal Printer is directly compatible with all the microcomputers in the Tandy range

A

ALU

The Arithmetic Logic Unit forms the heart of any computer that follows the von Neumann architecture — in other words all existing home computers. The ALU is a part of, and therefore found within, the CPU (Central Processing Unit), which on a home computer will be a 6502, Z80, or perhaps a Motorola 68000. While the rest of the CPU is concerned with shuffling data around the computer, at the right time and to the right places, the ALU is the only part of it that can actually change the data.

As the name suggests, the ALU can perform elementary *arithmetical* and *logical* functions on the data. Addition and subtraction are possible, as are functions such as AND, OR and NOT, but only on one byte at a time. The ALU performs its processing in the CPU's internal registers, in particular the accumulator (see page 13).

AND

AND is one of the fundamental logical functions, with which readers of THE HOME COMPUTER ADVANCED COURSE will already be familiar. AND takes on two arguments, each of which may be true or false — usually represented by binary 1 and 0 respectively. If, and only if, both inputs are true then the output or result of the AND function will be true also.

An AND gate is a hardware circuit that implements the AND function. A typical home computer will contain many of these, either in the form of dedicated logic chips, or integrated into the design of larger chips such as the microprocessor. But the AND function can also play an important role in programming, both in BASIC and machine code; in both cases it performs the function on several pairs of bits simultaneously. In machine code there will normally be an opcode for ANDing the contents of the accumulator with another byte in memory. In the resulting byte, a bit will have the value 1 if, and only if, the corresponding bits of the original bytes were both 1.

Most BASICS will allow AND to be used in a similar way, so that:

```
LET C=A AND B
```

will cause the computer to perform an AND on the binary representations of A and B, and store the result in the variable C. So if A is 13 (1101) and B is 11 (1011), C will take on the value 9 (1001). There are many applications for such a construct. For example:

```
IF (TURN AND 1)=1 THEN GOTO 1000
```

will only jump to line 1000 if TURN contains an odd (binary) number.

APL

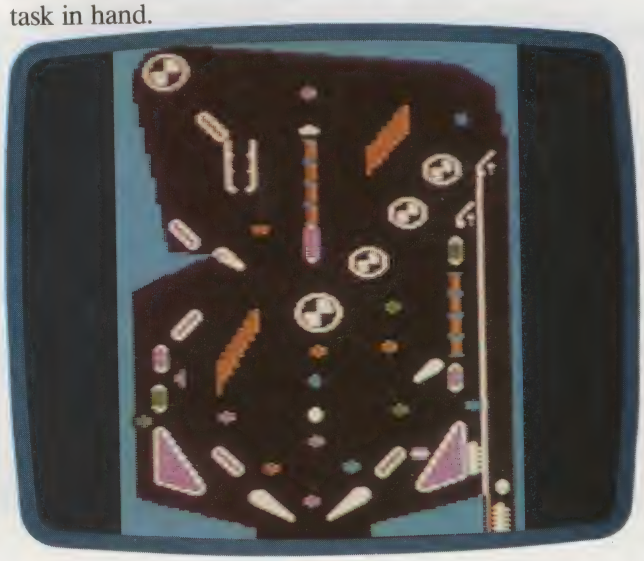
Computer jargon is full of strange acronyms, but they don't come much stranger than this one. APL is a programming language and it stands for A Programming Language! Despite the simplicity of its name, it is a very sophisticated, high-level language, and its users boast that they can program in a single line of APL what could take dozens of lines in BASIC. Though favoured in some academic and mathematical circles, its appeal is very limited, and it is consequently not generally available for microcomputers. Another reason for its unpopularity is that APL requires special symbols that are not available on most keyboards and screens. One microcomputer that could run APL was the SuperPET (a modification of the Commodore 8000 PET), which possessed a different processor (the 6809), more RAM and a character generator capable of producing the APL symbols on the screen.

APPLICATIONS GENERATOR

The idea of a program that can accept as its input a user's specification, and produce as its output a freestanding program that will perform the required task is a very appealing one — and not at all beyond the bounds of possibility. However, most of the commercial packages sold as 'program generators' are severely limited in scope, and produce programs that are considerably less efficient than code written by humans.

An *applications generator* is a similar but more popular idea. The applications programs that it produces are not freestanding — the original generator program is needed to run them. However, it does allow you to produce programs that are on the one hand tailor-made to your specifications, but on the other hand use efficient code that was originally written by a human programmer.

They work on the principle that within a particular area of application (say adventure games or book-keeping) all programs have to perform a minimum set of functions, which the generator provides ready-written. In specifying the exact operation of the required program, all the user is doing is creating a file of reference data that tells the applications generator how to apply its standard routines in the main program to the task in hand.



IAN McKINNELL

A WORD TO THE WISE

The prime requirements of a computer that is to be used for word processing are the quality of the keyboard and of the display. On both counts, the BBC Microcomputer scores highly. The two most popular software packages are available in Read Only Memory, which cuts out loading time and makes them very fast in operation.

One of the most popular word processing packages for the BBC Micro is Wordwise, which is stored on a chip that fits into a ROM socket in the computer. Once fitted, key in *W and the BBC becomes a word processor. The Wordwise package as delivered comprises the ROM chip, fitting instructions, a 30-page manual and a cassette that displays the text of the manual on the screen and usefully illustrates the functions of various commands.

Wordwise is straightforward to use and can be put to work almost immediately without prior training and constant reference to the manual. To get started you simply choose the edit mode by pressing the Escape key, and start typing. Editing is carried out by the cursor and letters can be keyed in or deleted. Wordwise makes use of the BBC function keys for moving and copying text.

As the user becomes more familiar with the greater flexibility that a word processor offers over the typewriter or pen and paper, more of the features can be explored. Jumping to the beginning or end of a line or page by using the Shift and Cursor keys, or overwriting letters by use of a function key, soon become part of the user's repertoire.

Characters appear on the screen in the 40-column mode and are clearly legible on a domestic television set. The final appearance of the text can be previewed on the screen in the 80-column mode by use of a function key. This shows how the text will appear as printed. It is probably at this preview stage that the newcomer to Wordwise first considers the general layout and appearance of the text and needs to refer to the section in the manual that covers embedded commands. These commands deal with the presentation side of word processing. An example is the command LM

Setting Up

The well-equipped word processing set-up should include keyboard and monitor (or television), disk and drive for storage, and printer and stationery for a hard copy print-out of text

KEVIN JONES

LIZ HEANEY

Alternative View

Acorn's own processing package for the BBC Microcomputer, View, is also delivered as a Read Only Memory chip. Like Wordwise, it can be installed so that it self-loads whenever the machine is switched on, or it can be entered from BASIC.

View uses the power of the BBC Micro's video generator to offer 76, 74, 34, or 16-column displays, the character size changing to suit. This effect is produced by 'magnifying' the entire screen image; the screen then becomes a window that travels across the much larger virtual screen. The Mode command, which effects these changes, is one that operates in View's Command mode, as do the disk access commands, find-and-replace, and print commands.

View's find-and-replace facility, in addition to the normal function of searching through the document for every occurrence of a given word, allows what the manual describes as a 'wild search'. The word is specified in the normal way, but ? is substituted for any doubtful character. Hence, if the key word is entered as th??, the search will find that, then, them, they and so on, — in fact, every four letter word beginning with th. Of course, it is necessary to be rather careful when defining the wild card search key, or the exercise can become self-defeating. One other member of the command set, especially useful to journalists who are often paid by the number of words written, is the package's ability to count words in a piece of text. All too often,word processing packages only keep a tally of characters.

View's other operating protocol, known as Screen mode, is divided into two groups of commands: Immediate, which controls character insertion/deletion, cursor movement, block movements and the other 'one-off' requirements; and Stored, which looks after the parameters in the creation or editing of a document, page formatting, headers and footers and other continuing requirements. In Screen mode, View makes excellent use of the BBC Microcomputer's function keys, each of the 10 having three distinct uses depending on whether it is used alone or in conjunction with the Shift or Control keys

followed by a number, which sets the position of the Left Margin. Similarly LL for Line Length, IN for INdent, PL for Page Length and so on. Being mnemonic, these commands are easily remembered and entered into the text after pressing a function key. They do not appear in the previewed or printed text.

If embedded commands are not used, Wordwise defaults to sensible values: 70 characters for line length, 66 lines for page length and five spaces for the left margin. Thus a well laid out document can be provided even before you master the formatting commands.

Control codes for the printer to select italics, emphasise characters, double the size of the characters and other options may also be keyed in as embedded commands. We will see later how function keys can be re-defined to facilitate the entry of printer control codes.

Wordwise operates in a menu mode to save, load and print complete sections of text. The menu appears on the screen when Wordwise is selected and displays eight options. The first four options are concerned with loading and saving text onto, or from, disk or cassette.

Option five allows the user to search for and replace specific items of text. An example in an article such as this would be to replace the word 'type' with 'key in'. The choices in this option are either *global* or *selective*: the former enabling the replacement to take place for all occurrences in the text; while the latter allows the user to move the cursor to the first occurrence, then the next and so on, choosing whether to replace the word at each occurrence. Menu option six allows the text to be printed and option seven allows it to be previewed. Option eight saves the formatted text without embedded commands.

To change from menu to edit mode, you press the Escape key and the text appears with the cursor positioned where it was left. Because the screen displays text in the 40-column mode, it is often necessary to hop from edit mode to menu option seven (preview) when formatting the text. One slight problem is that the effect of tabulation markers is not shown on the screen and frequent previewing of the text is often required to check the final layout of the document when printed.

One interesting aspect of the BBC is that the user-defined function keys used by Wordwise to move and copy or delete text and perform other tasks can be re-defined for use with the CTRL and SHIFT keys to, for example, produce a new paragraph, delete a complete line or include a printer code as an embedded command. Used on their own, the function keys retain the Wordwise defined functions. The BBC star (*) commands are also handy. These can be used in the menu mode to select a printer type, select tape or disk or return the machine to BASIC by keying *B.

Wordwise is undoubtedly a useful addition to any BBC Micro and is simple to install and use. There are more expensive and sophisticated word processors available for the BBC, such as View, but Wordwise would seem to cover most requirements, and although it can store only about 4,500 words as a single document, much more can be stored externally on cassette or disk.

TEXTUAL ANALYSIS

Before going on to investigate how machine code programs work, it is salutary to look at how BASIC programs are stored (in the BASIC Text Area of memory) and implemented (using the BASIC Interpreter program). This will serve as a reference point later when we come to discuss the way machine code operates in memory.

When you type or LOAD a BASIC program into the computer, you probably imagine that the computer is an empty vessel doing nothing until your instructions arrive. In fact, from the moment that the power is turned on, the computer is constantly running a sophisticated program of its own — the Operating System. This is a program, or set of programs, permanently burned into some of the ROM chips inside the machine. Its purpose is to make the machine work: it puts a display on the screen, it communicates with the printer and the disk drives, it scans the keyboard for keypresses, and so on. To the O.S. everything that comes into the machine is just data to be processed by its own programs.

One of these programs is called the BASIC Interpreter, and its purpose is to inspect the text of BASIC programs, and to implement their instructions. Everything in a BASIC program, therefore, is just data for the Interpreter program to process. When you type in a program, the Operating System recognises it as such because each new line begins with a valid line number. With some exceptions every character of that program line is stored in its own byte of the BASIC Program Text Area of memory. When you type RUN, the Operating System hands over control to the BASIC Interpreter, which — like any program — goes to work on processing its data (the contents of the BASIC Text Area).

The Interpreter does not change your program in any way, but simply interprets and implements it. And because the Interpreter obeys commands without question, it is quite possible to instruct it to look at the contents of any area of memory. If your program happens to allow you to inspect memory and you use it to inspect the Text Area, that's no paradox to the Interpreter. It just follows instructions if it can, and reports SYNTAX ERROR or OVERFLOW ERROR or something similar if it can't. It has neither the reasoning nor the vocabulary to issue error messages such as: TEMPORAL PARADOX or PHILOSOPHICAL DISCONTINUITY.

The Operating System stores your BASIC program character-by-character, with the exception of the BASIC keywords. Whenever it recognises the letters (or characters, or numbers, or voltage patterns) that make up a BASIC keyword, the Operating System replaces that word by a single-byte code number, called a *token*. This saves memory space — RESTORE, for example, would otherwise use up seven bytes — and means that the Interpreter's job of translating the BASIC program is much easier to perform.

Different machines use different token conventions, but, in general, token codes are numbers greater than 127. The ASCII codes for the printable characters (shown in the table on page 77) are all in the range 32 to 127. Therefore, any byte in the BASIC Text Area containing a number bigger than 127 must be a token byte put there by the Operating System. When the Interpreter encounters such a byte it simply implements the appropriate built-in subroutine.

The question arises, however, of why, when you LIST a program, you don't see unprintable characters, but rather the BASIC keywords, etc.? The answer is that during a LIST the Operating System inspects each byte of the Text Area, and whenever it finds a byte having a value greater than 127 it treats it as a token. Somewhere in memory is stored a complete list of the ASCII representations of BASIC keywords and the value of a token will point to that position. It's just the same as if the Interpreter were using the token's value to locate its implementation subroutine. And consequently, the Operating System puts the keyword rather than the token on the screen during a LIST. You can demonstrate this to yourself on a Commodore 64 very easily. (It's less straightforward on the BBC and Spectrum.) In lower-case mode, type:

```
100 rem*******h**********
```

Now LIST 100, and you should see:

```
100 rem*******left$**********
```

On the Commodore machines, the ASCII value of 'H' in lower-case mode is 200 so when the O.S. found a value of 200 in that particular byte during the LIST, it interpreted it as the token for the keyword LEFT$. If you now type:

```
100 rem"*******H**************"
```

and LIST 100, you'll see:

```
100 rem"*******H**************"
```

This demonstrates that it is important to remember that some printable characters, usually graphics characters, do have ASCII codes greater than 127, and they will be recognised as such,

provided they're in quotes. If not, they will be treated as tokens.

We are now at a point where we can begin to investigate how a BASIC program line is stored in memory. Computers differ in detail, but in general the first three or four bytes of a BASIC program line in the Text Area will contain the program line number and some information about the length of the line (see the panel). The line number that you attach to the line when you type it in is stored (although not as its ASCII equivalent — that would mean that line 61030 would require five bytes just to store its line number). Instead, the number is always stored in two-byte integer form. In this form, the numbers from 0 to 255 (which can be stored in one eight-bit byte, remember) are stored as a zero-byte followed by the byte containing the number. Numbers greater than 255 are stored exactly as paged addresses (see page 36): the value of the first byte is multiplied by 256, and added to the value of the second byte. 1000, for example, would be stored as 3,232 (3*256 + 232 = 1000). These two bytes are always in the same position in any line stored in the Text Area (although whether they are always the first two bytes or whatever depends upon the machine).

The information about the length of the line is placed in a single byte in the BBC and two bytes in the Spectrum. This represents simply the number of bytes in the line (including the two bytes for the line number and the line-length byte itself). If you know the address of the first byte of a BASIC program line in memory, and you add to it the contents of the line-length byte, then you will have the address of the first byte of the next program line. Since the biggest number expressible in one byte is 255, the maximum length of a BASIC program line on the BBC is, therefore, 255 characters. You might use the Mempeek program on page 59 to establish whether that is the limit of the number of *characters* you can type into a program line, or whether it is the limit of the length of the line as it is stored in the Text Area.

On the Commodore the line-length byte is replaced by two bytes called the *link address*. This is simply the actual address in two-byte form of the first byte of the next program line.

It's interesting to note that in the BBC and the Spectrum the next-line Start Address is calculated from the present address plus the line-length (which is slow but saves a byte); whereas on the Commodore the next address is stored as such

Conversion Chart

The American Standard Code for Information Interchange gives a standard character code value for the numbers 0-127. The codes 0-31 do not return printable characters, but are used to send control signals to peripherals such as the screen and the printer. The meaning of these codes, therefore, varies greatly from one machine to another — as the chart shows. Some machines, in particular the Commodore and the Spectrum in our chart, leave many codes unused (signified here by a ●). The codes 32-127 return the printable characters, and the standard ASCII codes in this range are common (with minor variations) to most computers. Your User Manual will give ASCII codes for your machine

ASCII CODE	ASCII	COMMODORE	SPECTRUM	BBC Micro
0	NUL — Does nothing	●	●	Null
1	SOH — Start heading	●	●	Next character to printer
2	STX — Start of text	●	●	Enable printer
3	ETX — End of text	●	●	Disable printer
4	EOT — End of transmission	●	●	Separate text/graphics cursors
5	ENQ — Enquire	White clr.key	●	Join text/graphics cursors
6	ACK — Acknowledge	●	PRINT	Enable VDU drivers
7	BEL — Ring bell	●	EDIT	Make short beep
8	BS — Backspace	Disables CBM key	Cursor left	Backspace cursor
9	HT — Horizontal tab	Enables CBM key	Cursor right	Forwardspace cursor
10	LF — Line feed	●	Cursor down	Cursor down
11	VT — Vertical tab	●	Cursor up	Cursor up
12	FF — Form feed	●	Delete key	Clear text area
13	CR — Carriage return	RETURN	ENTER	Return
14	SO — Shift out	L/case on	Number	Page mode on
15	SI — Shift in	●	●	Page mode off
16	DLE — Data link escape	●	INK	Clear graphics area
17	DC1 — Device control 1	Cursor down	PAPER	Def text colour
18	DC2 — Device control 2	Reverse on	FLASH	Def graphics colour
19	DC3 — Device control 3	Cursor home	BRIGHT	Def logical colour
20	DC4 — Device control 4	Delete key	INVERSE	Restore default log.clr
21	NAK — Negative acknowledge	●	OVER	Disable VDU drivers
22	SYN — Synchronous idle	●	AT	Select screen mode
23	ETB — End of transmission block	●	TAB	Reprogram display character
24	CAN — Cancel	●	●	Def graphics window
25	EM — End of medium	●	●	Plot m,x,y
26	SUB — Substitute	●	●	Restore default windows
27	ESC — Escape	●	●	Null
28	FS — File separator	Red clr.key	●	Def text window
29	GS — Group separator	Cursor right	●	Def graphics origin
30	RS — Record separator	Green clr.key	●	Move text cursor
31	US — Unit separator	Blue clr.key	●	Move text cursor to x,y

ASCII CODE	ASCII	ASCII CODE	ASCII
32	Space	80	P
33	!	81	Q
34	"	82	R
35	#	83	S
36	$	84	T
37	%	85	U
38	&	86	V
39	'	87	W
40	(	88	X
41	)	89	Y
42	★	90	Z
43	+	91	[
44	.	92	`
45	-	93	]
46	.	94	^
47	/	95	—
48	0	96	ı
49	1	97	a
50	2	98	b
51	3	99	c
52	4	100	d
53	5	101	e
54	6	102	f
55	7	103	g
56	8	104	h
57	9	105	i
58	:	106	j
59	;	107	k
60	<	108	l
61	=	109	m
62	>	110	n
63	?	111	o
64	@	112	p
65	A	113	q
66	B	114	r
67	C	115	s
68	D	116	t
69	E	117	u
70	F	118	v
71	G	119	w
72	H	120	x
73	I	121	y
74	J	122	z
75	K	123	{
76	L	124	!
77	M	125	}
78	N	126	"
79	O	127	Delete

(which uses an extra byte but is fast). This demonstrates that there's no *right* way to build a computer, there's only the individual designer's way. It is also a good example of the sorts of things that computer designers must take into consideration. They know that they have a fundamental choice between designing a machine that is slow but inexpensive, or one that is fast but expensive. Similarly, when writing a BASIC program on machines with limited memory (the unexpanded Vic-20 and ZX81 are good examples), you have to decide how to trade speed of execution against efficiency of memory usage.

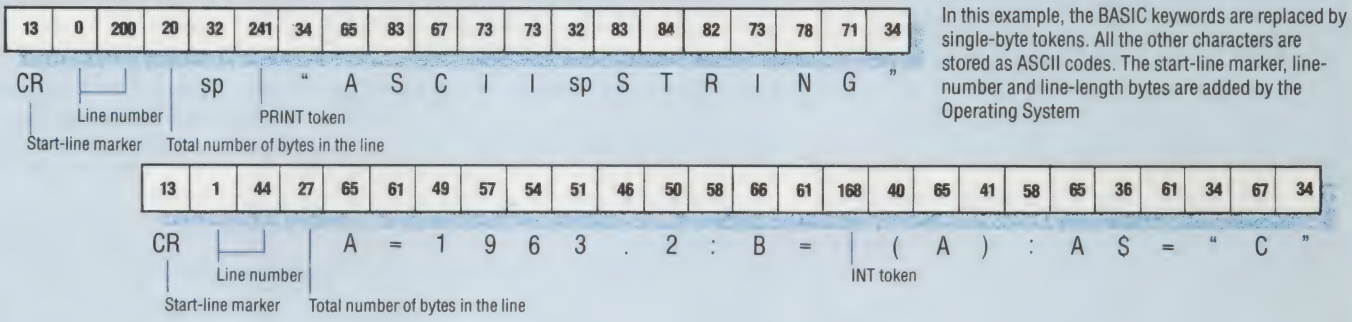
Finally, notice that in the Text Area there will be a line-start or line-end marker for each line of the BASIC program. On the BBC Micro each line starts with a byte containing 13 (ASCII for Carriage Return), whereas this ends a Spectrum line. The Commodore BASIC line ends with a zero-byte (ASCII for "").

How BASIC Programs Are Stored

Each machine has its own variations in the way it stores a line of BASIC Text. Consider the individual techniques shown for the lines of text given below

BBC Micro

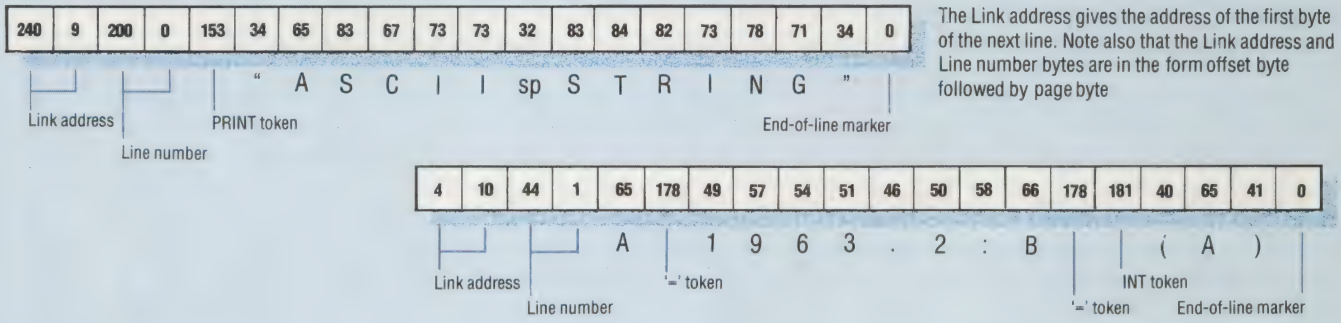
200 PRINT"ASCII STRING"
300 A=1963.2:B=INT(A):A$="C"



In this example, the BASIC keywords are replaced by single-byte tokens. All the other characters are stored as ASCII codes. The start-line marker, line-number and line-length bytes are added by the Operating System

Commodore 64

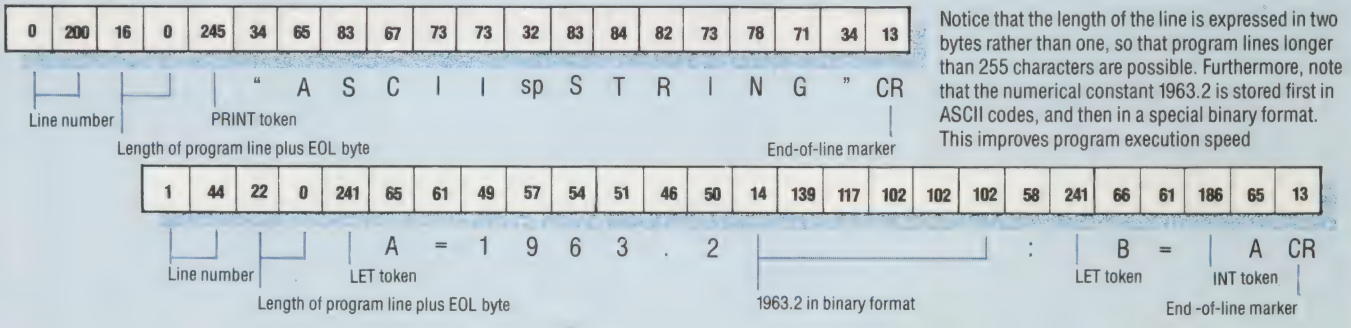
200 PRINT "ASCII STRING"
300 A=1963.2:B=INT (A)



The Link address gives the address of the first byte of the next line. Note also that the Link address and Line number bytes are in the form offset byte followed by page byte

Spectrum

200 PRINT "ASCII STRING"
300 LET A=1963.2:LET B=INT A



Notice that the length of the line is expressed in two bytes rather than one, so that program lines longer than 255 characters are possible. Furthermore, note that the numerical constant 1963.2 is stored first in ASCII codes, and then in a special binary format. This improves program execution speed

SOFT SELL

Imagine Software is one of the new generation of software houses riding high on the wave of enthusiasm generated by the booming sales of home microcomputers. The company is relatively new, having been established in late 1982, but now employs more than a hundred people — over 40 of whom actually write programming code.

As with all successful companies, Imagine has attained prominence in its particular field because of a blending of talents. The founders, David Lawson and Mark Butler, both Liverpudlians, represent the two essential elements in any business endeavour — technical expertise and business acumen, respectively.

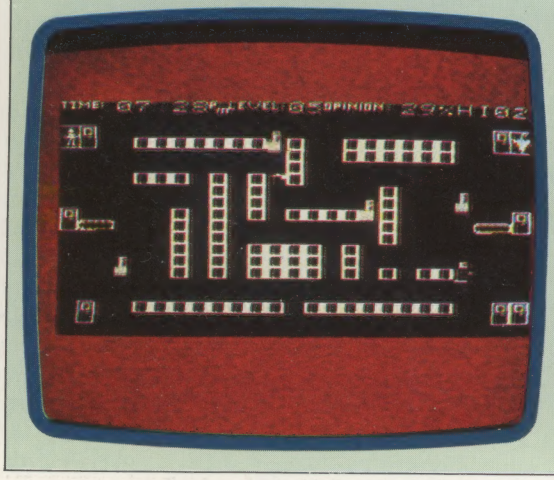
Imagine's speciality is in writing software for the range of Commodore machines. At one point four separate games programs — Bewitched, Catcha Snatcha, Wacky Waiters and the long-running Arcadia — written by Imagine for Commodore were in the Top Ten best-selling programs. Imagine's efforts with Commodore computers led to the unusual distinction of an invitation in 1983 to tour their Norristown, Pennsylvania factory and offices in order to preview the 264 and V364 machines. These machines are aimed at games enthusiasts, and so far Commodore has resisted the temptation to push its products up the price scale to attract the small business user. As a result of the tour, Commodore commissioned two games from Imagine, which will sell under their own imprint — a feather in the cap for such a young company.

In Britain, however, it is the Sinclair Spectrum that enjoys the widest software support. But Imagine has not always had the best of relations with Sir Clive Sinclair. When Mark Butler and Dave Lawson left Bug-Byte Software, where they spent their formative years, they paid a visit to Sinclair but left without coming to any working agreement. Imagine's policy is that its existing games can be modified for a new machine without anything much more complex than changing the memory locations of a few chunks of program code. This policy should certainly be tested by Sinclair's QL.

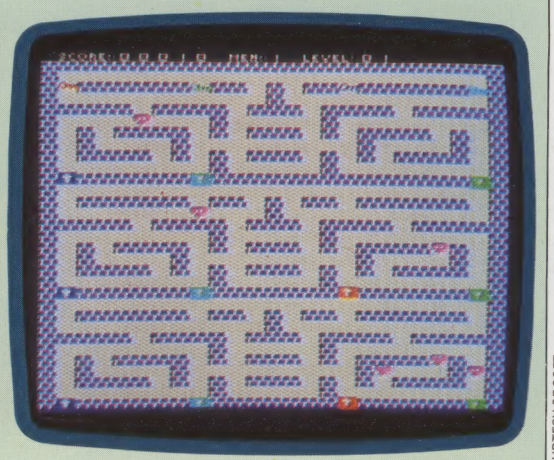
The QL is based on the Motorola 68000 central processor, and at present most programmers with a working knowledge of this chip are already employed by professional software firms or are in the computing departments of universities. Psion, the software house responsible for the development of the four programs that accompany the QL, had the benefit of a year's advance notice of the specifications of the computer, and was able to model its workings on a Vax minicomputer.

In order to find programmers to work with the 68000 processor, Imagine started advertising to fill a large number of posts, but on the whole results were disappointing. Though the response was enormous, few had more than a few months' experience. Many had not even succeeded in completing the coding of a single game. Not the

Catcha Snatcha
Written for the Commodore Vic-20 this particular maze/chase game sets a store detective to catching thieves, finding lost property and restoring wandering children to a parent



Bewitched
Another maze game, but different in that parts of the maze are accessible only after the doors into them have been successfully unlocked. To complicate matters further, not all the doors actually open. The placing of the doors is varied randomly, and there are ghosts to be avoided



COURTESY OF SOFT

raw material with which to start a new generation of programming projects!

Imagine now seems to be planning a new move, into the more lucrative (and more competitive) business software market. This was not so much a planned move as Apple actually made the first contact. Although at first surprised by this initiative, Imagine was quick to realise the direction in which Apple was pointing its software, with easy-to-use screen presentation and pictorial images such as 'windows' and 'icons' linked with the mouse input device, closely paralleled much games software, where graphic action and non-keyboard input are key factors.

Imagine had a thorough grounding in Apple's eight-bit technology, because it had used Apple IIe computers for writing programs from the outset. It was, therefore, ideally placed to take advantage of this contact. Also, as part of a retooling exercise, Imagine invested in several Sage IV machines. The object in this was to increase speed and computing power. Dave Lawson has written some original software, which is, in turn, used to write games programs, which are then cross-compiled (compiled on one machine for use in another) for the 'target' home computer. The Sage IV has a high-capacity RAM disk, which is invaluable when writing assembler. It is more economical of programmers' time to spend money on a machine like the Sage, rather than have programmers sitting around waiting for compiled programs. Supplied with the Sage is the UCSD (University of California at San Diego) p-System, a PASCAL implementation that enables it to mimic Apple's Lisa and Mackintosh, which both use PASCAL in the background. The terms 'background' and 'foreground' refer to machines that can run more than one program at a time. The foreground program always takes priority, but whenever an opportunity occurs, the background program will be run.

Imagine That
The strength of any software house relies on the relative input of the program design team. Brought together specifically to create the games Psyclapse and Bandersnatch, they are from left to right: Ian Weatherburn, Mike Glover, John Gibson and Eugene Evans

COURTESY OF IMAGINE

Like many software houses, Imagine has been investigating the most concisely named computer language — C. This is one of the most versatile and, more importantly, portable languages available for microcomputers. Its modular structure makes it ideal for developing systems software.

At present there are not many people around with experience of both C and the Sage microcomputer. As a result, Imagine is currently trawling colleges and other software houses in the hope of finding more expertise in this field.

THE WINNING FORMULA

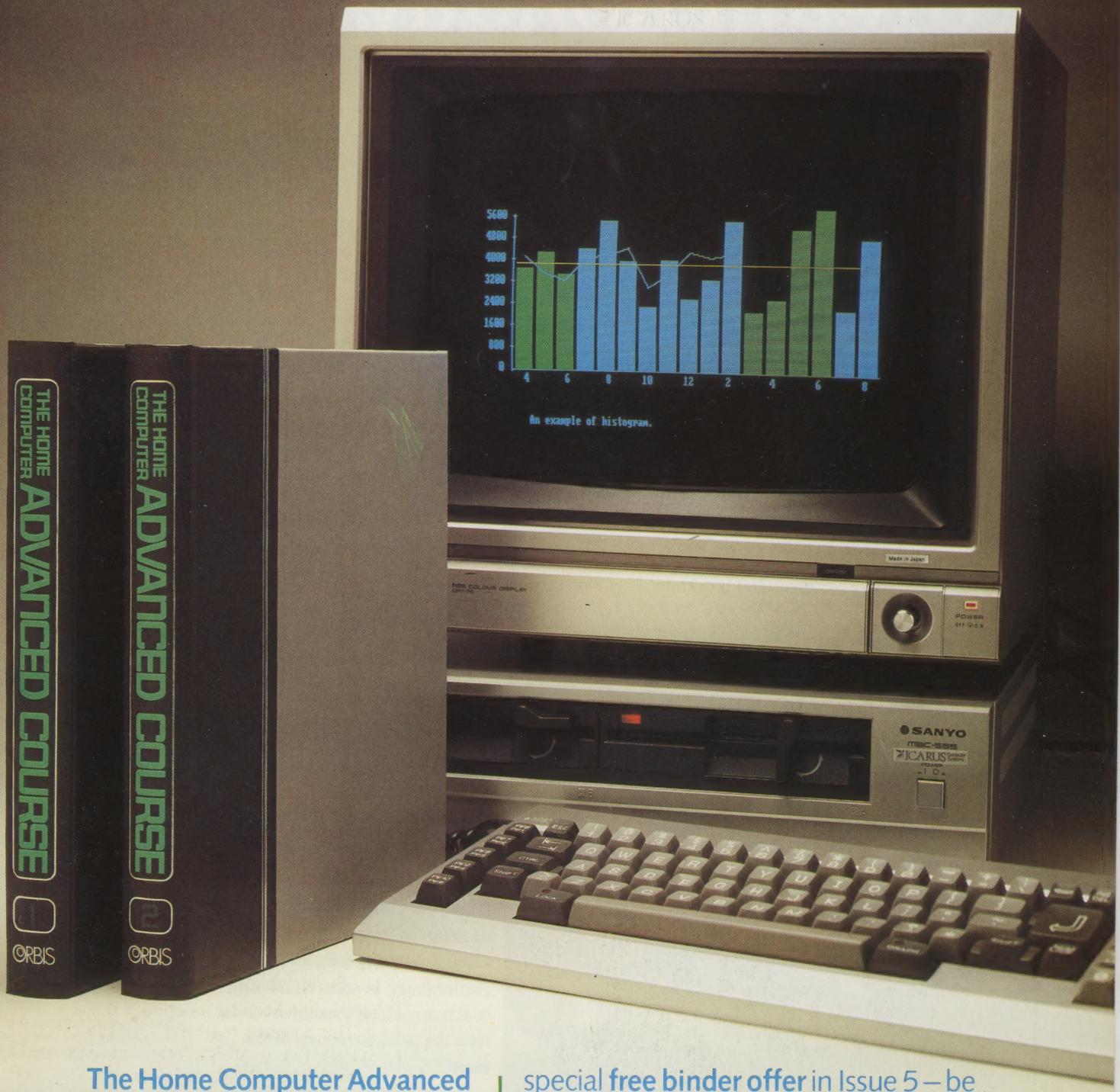
What is it that makes a piece of software a hit? A lot of it has to do with 'gut feeling'. In the beginning, when Mark Butler and Dave Lawson joined forces to launch Imagine, they already had a good idea of what their former employers, Bug-Byte, were successfully marketing. They would sit down to 'brainstorming' sessions — trying to think themselves into the mind of the games fanatic. This is vital. Considerations such as 'If this were my first computer, which I'd had for a month, or six months, what would I want from this game? Why should I play it? How long would I play it? Would I like the various sound effects or a graphics flash at this point in the play?' are all part of the developmental process. In effect, the games designer has to think himself down to the age of the intended customer.

Imagine is now too big for two people alone to devise all the games ideas, and it now has a team of eight graphics designers working up animations and storyboarding new plays. These are then tested by in-house staff — not the programmers. At this point in development you don't need the 'clutter' of technical considerations.

Unhappily for the software houses, one technique that all too many non-technical people have learnt is how to avoid paying for software. Illegal duplication is a very real problem. It has been estimated that for every cassette game sold over the counter, seven illegal copies are made, the vast majority by the simple expedient of audio (tape-to-tape) transfer. Preventative measures can be taken to eliminate or, at least, reduce this practice, but they are expensive — and this extra cost would have to be passed on to the consumer. Imagine is, no doubt, losing revenue this way, but it is still selling a great many cassettes, and for the moment is keeping its options open.

The cassette may not last too much longer before the cartridge and the disk replace it completely. Wholly new methods of software distribution may even take over, where program code is sent from a central, or host, computer and loaded into the home computer via the telephone lines that will service cable television. A software house that intends to grow in the present decade needs to be aware of such changes, and in this respect Imagine's Mark Butler is keeping his wits about him, hinting at new games possibilities using speech synthesis and keeping abreast of innovations in technology, such as the laser disk.

The volume 1 binder can be yours free!



The Home Computer Advanced Course will take you far beyond the novice stage, widening your knowledge and making you a more sophisticated user.

To help you keep your copies immaculate, we will be making a very special **free binder offer** in Issue 5 – be sure not to miss it!

Overseas readers: this special offer applies to readers in the U.K., Eire and Australia only. Binders may be subject to import duty and/or local tax.

A dream computer, A dream holiday,

and a sight for twenty sore eyes.

WIN – 1st Prize, an I.B.M. P.C. system to the value of £5,000.

WIN 2nd Prize, a holiday for two in the U.S.A. with a visit to Silicon Valley.

WIN – 3rd Prize, one of ten high resolution colour monitors.

ENTER TODAY

ALL YOU HAVE TO DO

"AH! DRESS THE APE RIGHT,"
EXCLAIMED AMOS, BUGLE SALES MANAGER.

This crazy sentence contains anagrams of four familiar pieces of computer jargon.

For the last three weeks we have been featuring the particular parts of the above sentence that make up one of the four words or phrases.

This week (issue 4) the final well-known computer word or phrase is hidden in the words "BUGLE SALES MANAGER"

HOW TO ENTER

Put your answers on a postcard or the back of a sealed envelope as follows:

Your name _____

Your address _____

_____

Telephone number _____

Hidden words/phrases 1. _____

2. _____

3. _____

4. _____

Your invented piece of computer jargon (5 words or less) _____

_____

and its definition (15 words or less)

_____

_____

Send your postcard/sealed envelope to Dept. H.C.A.C. Orbis Publishing Ltd., 20-22 Bedfordbury, London WC2N 4BT to arrive not later than three weeks from the publication date of this issue.

RULES

1. By entering the competition, competitors will be deemed to have accepted and agreed to abide by the rules.
2. The competition is open to all UK and Eire readers other than employees or their families of Orbis Publishing Ltd. and their advertising and servicing agents.
3. All valid entries will be examined.
4. The first prize will be awarded to the competitor who selects, in the judges' opinion, the most suitable answers to the questions listed. In the event of more than one competitor qualifying for the first prize this will be awarded to the qualifying competitor who, in the judges' opinion, submits the most apt answer to the special tie-breaker question above. A similar basis will be adopted for determining the winners of the runner-up prizes. No household may win more than one prize.
5. All entries must be in ink or ball-point pen on a postcard or sealed envelope.
6. The closing date will be three weeks after publication of this issue. Winners will be notified by post and a full list of winners will be available by post on supply of a S.A.E. one month after the closing date.
7. No responsibility will be taken for entries lost, delayed or damaged in transit. Proof of posting cannot be accepted as proof of delivery.
8. Illegible entries and entries not made in accordance with the rules and directions will be disqualified.
9. All entries submitted will become the copyright of Orbis Publishing Ltd. and no entries can be returned.
10. The judges' decision is final and legally binding, and the decision of Orbis Publishing Ltd. on all other matters concerning the competition will also be final and legally binding. No correspondence will be entered into.

This competition is open to readers in the UK and EIRE only.